METHOD FOR QUICKLY BOOTING A COMPUTER SYSTEM

BACKGROUND OF THE INVENTION

1. Field of the Invention

The present invention relates to a method of and apparatus for booting a personal computer system and, more particularly, for quickly booting a computer system, in which a boot configuration information is created and saved in a disk for future boot, and the saved boot configuration information is reused upon the request of the subsequent boot.

2. Description of the Related Art

Sub A1

Figure 1 shows a block diagram of the system architecture for a conventional personal computer system, comprising a central processing unit (CPU) 1; a read only memory (ROM) 2 for

permanent storage of basic input output system (BIOS) and the initial states of internal devices, a random access memory (RAM) 3 for temporary storage of information; a micro computer (MICOM) 4 for controlling peripheral devices such as a keyboard input device, a mouse input device, and a power supply; a hard disk (HDD) 8 for providing a secondary information storage; a disk controller 5 for controlling HDD; a video output display 6 for displaying information; and a power supply 7. When power is applied to the computer system, the computer system starts to be booted to load an operating system (OS) and thus is brought into a known useful state in which application programs can be executed. This procedure is generally called "booting". An operating system is a software that provides resource management on a computer system, including basic tasks such as process execution, memory management, and file management. Examples are MS-DOS, Windows95, OS/2, and UNIX. Execution of user applications is based on these basic functions of the operating system.

A1

Sub A2

The boot process of an IBM PC in which MS-DOS operating system is already installed is as follows. When a user turns the personal computer power switch on or presses a reset button, a power-on self test (POST) is performed by ROM BIOS codes to diagnose each component of the personal computer. Next, a file called MSDOS.SYS is loaded and executed, and another file called IO.SYS is then loaded and executed to perform certain

3

A2

preliminary functions related to management of such peripheral devices as keyboard, disk, and display. And then, a command preprocessor or COMMAND.COM is loaded into a memory that receives, interprets and executes user commands. A file called CONFIG.SYS that specifies devices possibly connected to the personal computer is loaded and ASCII statements contained therein are executed to load device drivers and initialize them. Finally, another ASCII file called AUTOEXEC.BAT is loaded and then programs that is listed therein are executed, thereby preparing the personal computer for use.

Sub A3

There two kinds of boots; "cold boots" and "warm boots", which rely on the state of the computer system when the boot operation is requested. A "cold boot" is performed when power is applied to computer or a reset button is pressed. When an operating system is loaded in memory already and the computer system is powered on already, a user may request a "warm boot" by entering a predefined sequence of key strokes, e.g., <Ctrl>+<Alt>+<Del>. The BIOS codes consist of a plurality of computer routines for controlling devices such as system clock, video output display, disk driver, and keyboard and thus provide a low-level interface to these devices. The BIOS is generally stored in a Flash ROM.

Shortly after power on or a reset button is pressed, the CPU begins executing the ROM BIOS codes. The BIOS codes for POST are, first, executed to diagnose and initialize devices

4

attached to the computer system and obtain the status of the devices.

When a "warm boot" is requested or a reset button is pressed, it is desirable that the time required for the boot process is reduced to force the computer into a ready state as quickly as possible. The boot process is usually called "quick boot", which is achieved by simplifying some device diagnosis processes or loading the device status information that was obtained at the preceding boot time from a storage medium such as disk. Because the quick boot means a boot process in which some POST operations, e.g., memory test are skipped, the quick boot is generally referred to as "quick post".

FIG. 2 is a flowchart of the quick POST in an IBM personal computer system in which Windows95 is installed according to the conventional art. When the computer system is powered on or a reset button is pressed (S11), the Windows95 is loaded into a memory after execution of a normal POST process (S12). To be specific, once the POST process is performed, ASCII statements in CONFIG.SYS and AUTOEXEC.BAT are executed sequentially and WIN.COM is then executed to load Windows95. While Windows 3.1, a previous version of Windows95, is loaded after the personal computer is booted on the basis of MS-DOS, Windows95 installed PC is booted and Windows95 user interface is provided directly.

Once the boot operation is completed, a basic boot information is saved to a disk for future quick POST process

(S13). After that, if a user requests a "quick boot" to reboot the personal computer (S14), the above-mentioned quick POST process is performed to reduce the time needed to complete a normal POST process. As another method, the POST process execution is skipped by using a basic boot information that was created and saved in a disk immediately after the preceding POST process is completed.

However, the conventional quick boot relies on the POST process, e.g., the omission of memory test. In other words, in the conventional quick booting method, the same operations as those of normal boot process are still performed after the quick POST process. Therefore, in case where there are a lot of ASCII statements in CONFIG.SYS and AUTOEXEC.BAT, the quick boot of the conventional art is not effective to reduction of the boot time.

According to the conventional booting method, in Windows95 installed personal computer system, working environment or all information stored in memory are saved to a disk for the subsequent quick boot. If memory size is larger than 32 MB, the amount of data to be saved to the disk becomes too large. As a result, the subsequent booting by reloading the saved data into the memory may be even slower than a normal boot.

SUMMARY OF THE INVENTION

Sub A4 It is therefore a primary object of the present invention

A4

to provide a method and apparatus that significantly reduces the time required for boot process after a POST operation by using a boot configuration information on memory and the attached devices that was created and saved in a disk in the preceding boot process, and thereby skipping execution of statements in an initial device configuration file and an automatic batch file.

To achieve the object, the present invention provides a method for quickly booting a personal computer system, comprising the steps of performing a POST operation when the system is powered on or a reset button is pressed; checking if a boot configuration information that was created in the preceding boot process exists in a disk; saving the boot configuration information to the disk after execution of a POST operation on the basis of the checking result; and loading a graphic user interface (GUI) program.

The method for quickly booting a computer system according to the present invention is also characterized in that it comprises the steps of performing a POST operation when the system is powered on or a reset button is pressed; restoring a boot configuration information by using the boot configuration information that has been saved in a disk; and loading a GUI program.

According to the quick booting method of the present invention, after ASCII statements listed in an initial device

7

configuration file and an automatic batch file are executed, a boot configuration information that is resident in a memory, i.e., the status of devices and the contents of memory are saved into a disk. After that, when a reboot is requested, a computer system can be booted quickly by using the stored boot configuration information, without execution of the initial device configuration file and the automatic batch run file.

BRIEF DESCRIPTION OF THE DRAWINGS

The accompanying drawings, which are included to provide a further understanding of the invention, illustrate the preferred embodiment of this invention, and together with the description, serve to explain the principles of the present invention.

In the drawings:

FIG. 1 is the system architecture of a general personal computer system;

FIG. 2 is a flowchart showing the conventional method for quick POST operation in a Windows95-installed personal computer;

FIG. 3 is a flowchart showing a method for a quick boot according to an embodiment of the present invention;

FIG. 4 is a flowchart showing a method for saving a boot configuration information after execution of POST operation in a Windows95-installed personal computer according to an

embodiment of the present invention;

FIG. 5 is a flowchart showing a method for restoring a stored boot configuration information in a Windows95-installed personal computer according to an embodiment of the present invention;

FIG. 6 is a flowchart showing a method for saving the contents of memory into a disk according to an embodiment of the present invention; and

FIG. 7 is a flowchart showing a method for restoring the contents of memory according to an embodiment of the present invention.

DETAILED DESCRIPTION OF THE PREFFERRED EMBODIMENT

The preferred embodiments of the present invention will be described below in detail referring to the accompanying drawings.

FIG. 3 is a flowchart of the quick boot process in an IBM personal computer system according to an embodiment of the present invention. The flow proceeds to step S21, in which a POST operation is performed when a computer system is powered on or a reset button is pressed. A normal boot process of an operating system, e.g., Windows95 is then executed (S22). Next, a boot configuration information, i.e., the contents of memory and the status of the attached devices that was created and has been resident in a memory since execution of the POST operation

9

is saved to a disk (S23). A RAM-resident program is called by a software interrupt (INT in general) in modified ROM BIOS codes so as to save the boot configuration information into a disk for future boots. Since then, when a reboot is requested (S24), the POST operation is performed and then the saved boot configuration information is retrieved from the disk in order to complete the reboot process (S25). If it is determined that either CONFIG.SYS or AUTOEXEC.BAT was changed (S26), the changed two files are loaded into a memory and are then executed to form a new boot configuration information, which will be saved to the disk again for the subsequent boot. In this way, if CONFIG.SYS and AUTOEXEC.BAT are not changed, they do not need to be loaded and executed when a computer system is booted, resulting in a quick boot.

FIGS. 4 and 5 are flowcharts respectively showing methods for saving and restoring a boot configuration information in a Windows95-installed IBM personal computer according to an embodiment of the present invention.

The method for saving the boot configuration information to a disk will be described now in detail referring to FIG. 4. When power is turned on or a reset button is pressed (S31), a cold boot or warm boot is requested. The POST operation is, first, executed (S32) and then an INT 19h service routine is called to load an operating system (S33). By calling the INT 19h, control is passed to a bootstrap loader which loads the

10

operating system into a memory to prepare the personal computer for use.

Next, CONFIG.SYS is loaded into the memory and statements therein are executed. CONFIG.SYS includes ASCII statements describing the size of disk buffer, the number of files that can be opened simultaneously, the names of device drivers needed to control devices attached to the computer system, and so on. After executing CONFIG.SYS, another ASCII file called AUTOEXEC.BAT is loaded into the memory. The file names of programs that a user wants to run automatically at the boot time are listed therein, and the programs are executed (S34).

Next, a RAM-resident program is activated to replace an original INT 2Fh service routine in the ROM BIOS codes. To do this, the interrupt vector for INT 2Fh is substituted for the address of the RAM-resident program (S35). Next, WIN.COM is executed to load Windows95 into the memory (S36). The INT 2Fh service routine is called by using software system management interrupt (software SMI) during the execution of WIN.COM. At the interrupt point, the contents of a particular register is sent to the RAM-resident program and then performs a prescribed function associated with the register contents (S37).

If the register contents is a predetermined value, e.g., 1605H, the RAM-resident program checks if there is a file that contains the boot configuration information in a disk (S38) and saves the current boot configuration information to the disk,

11

if not (S39). WIN.COM is then executed to load a GUI program of Windows95 into the memory (S41), providing a user with Windows95 interface (S42). It should be noted that the boot configuration information is saved to the disk immediately before Windows95 loads device drivers into a memory, i.e., an extended memory is used to load GUI program of Windows95.

The operation of saving the boot configuration information to a disk (the step S39 of FIG. 4) is described in detail with reference to a flowchart of FIG. 6. The contents of memory block of a predetermined size are, first, examined and are then saved to the disk if the memory block is satisfied with a predetermined criterion. An address of the memory block is saved to the disk, as well. To be specific, if it is determined that the boot configuration information resident in a memory needs to be saved to the disk (S71), the INT 2Fh service routine checks if a memory segment of 64 KB is filled with '0', while scanning every memory segment (S72). If not, the contents of the memory segment are saved to the disk (S73), together with its address (S74). The memory segment is treated as a memory accessing unit, which is 64 KB in size in the IBM personal computer system. And the boot configuration information to be saved is approximately 7MB in size, which is composed of 1MB for saving the software SMI, 4MB for the video memory, and 2MB for saving a memory area in which the interrupt vector table and some crucial programs for system management are resided.

The next time the computer system is powered on or reset, the saved boot configuration information is used to boot the computer system. The method for restoring the boot configuration information will be described now in detail referring to FIG. 5.

Once power is turned on or reset button is pressed (S51), a quick POST operation including skip of memory test is executed (S52), and then it is checked whether or not there is any boot configuration information that has been saved to a disk in the preceding boot process (S52-1). If it is determined that a boot configuration information exists, the operation for its restoration is performed (S53).

The process for restoring the boot configuration information is described in detail referring to a flowchart of FIG. 7. First, it is checked whether or not a current boot configuration has been changed based on the restored boot configuration information. If there is any change in the boot configuration, commands that are usually executed at the boot time, for example, commands for initial setup device configuration are executed and then a newly formed boot configuration information is saved to the disk for future boot. Specifically, when a computer system is resumed, it is checked if the boot configuration information will be restored (S81). If it is determined that the boot configuration information is restored, the contents of memory segments, addresses of which

13

was saved before in the disk, are copied to the memory at their own addresses (S82). Other memory segments than the restored memory segments become filled with '0' (S83). The reason why the contents of those segments are not restored is that they are set to all '0's during the BIOS POST operation.

Once restoration of the contents of those memory segments is completed, it is checked if CONFIG.SYS and AUTOEXEC.BAT was changed (S54). If it is determined that either CONFIG.SYS and AUTOEXEC.BAT was changed, the bootstrap loader, the INT 19h service routine is called (S55), and then the both ASCII files are loaded into a memory to execute statements therein (S56). Next, the RAM-resident program is activated to replace the original INT 2Fh service routine (S57).

Next, WIN.COM is executed to load Windows95 into the memory (S58). The INT 2Fh service routine is called through the software SMI during the execution of WIN.COM (S59) and thus the RAM-resident program is executed. The RAM-resident program, first, checks the contents of a particular register and then, if it is matched with a predetermined value, saves the current boot configuration information that is resident in the memory to the disk (S61). The method for saving the contents of memory where the boot configuration information is resided is the same as that shown in FIG. 6. Next, control is passed to WIN.COM again and thus Windows95 GUI is set up (S62), thereby preparing the computer system for use (S63).